DATA PACKET COMMUNICATION DEVICE

BACKGROUND OF THE INVENTION

1. Field of the invention.

The present invention relates to network communication, and, more particularly, to a packet processing node for network communication.

2. Description of the related art.

In many communication systems, any transmission from one node is sent to a plurality of connected nodes. In such broadcast systems, the information may only be relevant to a single receiver, but all nodes must process the transmission to determine its intent. Data on such networks are usually broken into pieces known as packets. A packet is formed of consecutive bytes that may be marked by leading and/or trailing delimiters. Protocols by which nodes communicate usually divide a packet into the protocol-relevant information (the header) and the data relevant to the receiving node (payload).

It is known for the protocols to be "layered" so that the payload area of a lower protocol contains a packet of the next higher layer protocol. Most communication environments require that addressing information be contained at fixed locations (in the first few bytes) of the lowest layer protocol header so that hardware can help discard unwanted packets by simple filtering.

There are many examples of commercially available network devices that perform such filtering. Practically all Ethernet media access controllers (MAC's), for instance, provide for filtering based on an exact match of the first six bytes of a packet header to a programmed value (the network address). This greatly reduces the load of the general-purpose central processing unit (CPU). However, many network frames that are addressed to the node either distinctly or by a shared address are still not relevant to a particular node. The receiving equipment may be required to look at an entire packet to determine its relevance. Additional cost is inherent in networked systems because memory is required to store the entire packet while the processing elements perform the analysis.

Most of the existing patents relating to packet filtering are a result of the very competitive commercial network switch and router market. Routers must sort received frames into subsets based on different byte fields within a packet. To achieve high throughput, specialized hardware can be used to perform this sorting. In all known cases, an assumption is made that a CPU is present and only assists the microprocessor in performing its task. A problem is that a microprocessor adds significant cost to the product.

What is needed in the art is a device for network communication that does not require a microprocessor.

SUMMARY OF THE INVENTION

The present invention provides a device for network communication using specialized hardware that eliminates the need for a microprocessor in the system. The device processes information in a packetized communication system. The result is a minimal-cost packet-processing node.

A hardware apparatus extracts only pertinent information from received packets and can generate response packets based on the same information. This processing is done in real time and without the use of an additional data processing system that might include a microprocessor or storage memory.

The invention comprises, in one form thereof, a method of processing data packets. A plurality of the data packets are received at a selected node. Pertinent information in the data packets is extracted. The pertinent information is pertinent to the selected node. A plurality of response data packets are generated based on the pertinent information. The extracting and generating steps are performed without use of a microprocessor.

The invention comprises, in another form thereof, a data packet communication system including a peripheral device and a filter device connected to the peripheral device. The filter device receives a plurality of data packets and identifies pertinent information in the data packets. The pertinent information is pertinent to the peripheral device.

An advantage of the present invention is that network packet processing is performed without the need for a microprocessor.

Another advantage is that the packet filter hardware not only identifies packets relevant to the receiving node, but also extracts any pertinent information contained in the data stream.

Yet another advantage is that signals are created by the filter hardware to indicate that a response packet should be generated using the extracted information.

A further advantage is that the packet filter hardware can be replicated and used in parallel to provide different protocols within the same network node.

BRIEF DESCRIPTION OF THE DRAWINGS

The above-mentioned and other features and advantages of this invention, and the manner of attaining them, will become more apparent and the invention will be better understood by reference to the following description of embodiments of the invention taken in conjunction with the accompanying drawings, wherein:

Fig. 1 is a block diagram of one embodiment of a packet communication device of the present invention connected to a peripheral device and a packetized data network;

Fig. 2 is a block diagram of one embodiment of a data packet processed by the packet communication device of Fig. 1;

Fig. 3 is a block diagram of the hardware filter of Fig. 1;

Fig. 4 is a schematic diagram of the signaling logic of Fig. 3; and

Fig. 5 is a block diagram of the packet generator of Fig. 1.

Corresponding reference characters indicate corresponding parts throughout the several views. The exemplifications set out herein illustrate one preferred embodiment of the invention, in one form, and such exemplifications are not to be construed as limiting the scope of the invention in any manner.

DETAILED DESCRIPTION OF THE INVENTION

Referring now to the drawings and particularly to Fig. 1, there is shown one embodiment of a packet communication device 10 of the present invention. Device 10 includes neither a microprocessor nor storage memory. That is, device 10 is microprocessorless and memoryless. Device 10 includes a hardware filter 12 and a packet generator 14 connected to a packet data network 16. Hardware filter 12 and packet generator 14 are also connected via an interface 18 to a payload data consumer in the form of a peripheral device 20. An optional protocol state machine 22 may be necessary to convert filter indications into transmit requests to packet generator 14. Network 16 may be any of the common serial network architectures (such as Ethernet or Token Ring) or generally any packetized data stream. Peripheral device 20 can be any data consumer system that utilizes information, e.g., a computer, printer, set-top box, or other peripheral. Interface 18 may be in the form of a first in first out (FIFO) memory or data buffer.

The format of one embodiment of a network data packet processed by device 10 is shown in Fig. 2. Only fields of interest, i.e., pertinent, to the receiving node, i.e., peripheral device 20, are important and are therefore processed. The pertinent information is the minimum amount of information needed to correctly implement the protocol. That is, the pertinent information is what is mandatory for a minimal implementation of the protocol. Bytes of interest, i.e., pertinent data, include the payload data as well as header data that might identify the packet type, the intended receiver, sequencing information, and so forth. Such pertinent fields include a sequence number field, which indicates the order the data is in, a destination address (DA) and a source address (SA). Thus, information pertinent to the receiving node includes selected bytes within the data packets.

Other packet bytes not of interest to peripheral device 20, such as diagnostic information, are treated as "don't care" information and are ignored. Such non-pertinent fields include a checksum field, a priority field, start-of-packet delimiter (SOP) and end-of-packet delimiter (EOP).

Hardware filter 12 receives packetized data from network 16 and includes an offset counter 24 (Fig. 3), pattern generator 26, mask generator 28, multiplexer 30, maskable comparator 32, event decoder 34 and signaling logic 36. All of these blocks are easily constructed by someone skilled in the art. The incoming packet data stream is assumed to be a stream of words (of fixed bit-width W) with start-of-packet (SOP) and end-of-packet (EOP) indicators that are synchronous to a common receive clock. This is a common interface for media access control (MAC) hardware. The outputs of hardware filter 12 include an extracted payload data stream (payload word), protocol parameters, and indicators of a successfully accepted packet.

Offset counter 24 includes an N-bit counter that tracks the number of words from the start of a packet. Counter 24 is wide enough to accommodate a maximum-length packet for the particular network medium. Offset counter 24 is cleared upon receiving the SOP indication and increments by one with each packet word received.

Packet generator 26 is a special case of an N to M decoder. Given an input value of width N, pattern generator 26 outputs a predetermined corresponding word of width M. The decoder is designed so that, given the input packet offset, the output of the decoder is the corresponding word required for packet acceptance.

Mask generator 28 is similar to pattern generator 26. Mask generator 28 decodes offset counter 24 and produces a bit-mask. The mask indicates which bits within the word are compared. An additional set of outputs is routed to multiplexer 30. These outputs determine whether multiplexer 30 should pass the output of pattern generator 26 or one of the external comparison values to comparator 32. The external comparison values are specific to the receiving node (peripheral device 20), such as its network address, or are states within the protocol. For protocols where it is only necessary to examine entire words, only the multiplexer control outputs are necessary.

Maskable comparator 32 indicates whether the bits in the received packet work unmasked by mask generator 28 are equal to the corresponding bits in the output of multiplexer 30. A logical "0" on the output of comparator 32 indicates that all unmasked bits match. If the word is a don't care (all bits masked), the output is also a "0".

Event decoder 34 indicates when header information needs to be stored into memory elements (flip-flops). This allows protocol state machine 22 to use the information when updating its state or transmitting a response. Event decoder 34 is also used to generate a Payload Start (PS) signal to indicate where in the packet payload data should be extracted.

Signaling logic 36 generates indicators to outside entities. A signal, RELEVANT (Fig. 4), is generated with an R-S flip-flop 38. The start-of-packet (SOP) signal is connected to the "S" input of flip-flop 38, with the "R" input connected to the output of maskable comparator 32. Thus, the RELEVANT signal goes high at the start of a packet and remains high until comparator 32 finds a problem with the packet. The RELEVANT signal is logically AND'ed with the end-of-packet (EOP) signal to create a PACKET GOOD signal for use outside filter 12. Another R-S flip-flop 40 generates a PAYLOAD VALID signal. The PAYLOAD START signal is AND'ed with the RELEVANT signal, and the result is connected to the "S" input of flip-flop 40. The end-of-packet (EOP) signal is connected to the "R" input of flip-flop 40. This creates a signal that is active for the entire payload of packets whose header matches the acceptable criteria.

In operation, when the start-of-packet (SOP) indicator is given, offset counter 24 is cleared and the RELEVANT signal is set active. Packet data starts entering filter 12, with counter 24 incrementing with each word. At each word, mask generator 28 determines whether any part of the word is necessary for comparison. Mask generator 28 also determines which value should be used for the comparison, i.e., the output of pattern generator 26 or one of the external comparison values. The output of pattern generator 26 is a "don't care" condition whenever a comparison is not performed. Multiplexer 30 steers the appropriate compare word to comparator 32, and comparator 32 indicates if a necessary match is missing. If it is so indicated, the RELEVANT signal drops inactive and processing is ignored for the remainder of the packet, although the processing does continue. If the header is successfully processed, i.e., the PAYLOAD START signal is generated, then the PAYLOAD VALID signal activates and the packet payload data is passed to peripheral device 20 for consumption. This continues until the end-of-packet (EOP) signal is indicated, which causes the PAYLOAD VALID signal to deactivate and the PACKET GOOD signal to be pulsed.

Meanwhile, event decoder 34 extracts header data, e.g., extracted elements 41, by activating the enable inputs of storage flip-flops attached to the data stream. When the PACKET GOOD signal is pulsed, protocol state machine 22 can determine if a response is necessary. If such a response is called for, the extracted header information is then available to packet generator 14 as header input data for transmission.

Other embodiments of the hardware filter might be needed for difficult protocols. The comparator could include more arithmetic logic unit (ALU)-like functions (less than, greater than, etc.) than merely the "equals" operation. Pattern generator 26, mask generator 28 and event decoder 34 could be replicated and connected in parallel to identify various types of packets.

Multiple hardware filters 12 could be used in parallel to detect different protocols, or might be cascaded serially to allow for the stripping of the individual protocol layers, as needed.

Packet generator 14 transmits packets to network 16. As shown in Fig. 5, packet generator 14 includes the same building blocks as those found in hardware filter 12. Packet generator 14 includes an offset counter 42, a pattern generator 44, a multiplexer 46, an event decoder 48, and a mask generator 50.

Offset counter 42, pattern generator 44, event decoder 48, and multiplexer 46 perform the same functions as offset counter 24, pattern generator 26, event decoder

34, and multiplexer 30, respectively. Pattern generator 44 may or may not have the exact same patterns as pattern generator 26. Pattern generator 44 is protocol-dependent.

Mask generator 50 is only used to create the multiplexer steering controls. The masks need not be generated since no comparisons are performed on the transmit side of device 10, i.e., within packet generator 14

Packet generator 14 operates exactly as the logic used to create the comparison word in filter hardware 12. When a packet is to be sent, offset counter 42 is cleared and pattern generator 44, mask generator 50 and event decoder 48 decode the offset into a transmission word. This word is sent to media access controller 52 for transmission. When media access controller 52 requires the next word, offset counter 42 is incremented and the process continues until all words of a packet have been sent. Event decoder 48 creates the necessary framing signals SOP and EOP.

While this invention has been described as having a preferred design, the present invention can be further modified within the spirit and scope of this disclosure. This application is therefore intended to cover any variations, uses, or adaptations of the invention using its general principles. Further, this application is intended to cover such departures from the present disclosure as come within known or customary practice in the art to which this invention pertains and which fall within the limits of the appended claims.